Section 112, Second Paragraph, Rejection:

The Office Action rejected claims 19-23, 34 and 52 under 35 U.S.C. § 112, second paragraph as indefinite. Specifically, the Examiner stated that it is not clear whether the platform-independent programming language objects comprised in one or more libraries are the same platform-independent programming language objects referenced in the parent claim as storing logical commands.

Applicant's 19 recites in pertinent part:

> modifying one or more platform-independent programming language objects comprised in one or more object libraries;
> wherein the modified one or more platform-independent programming language objects comprised in the one or more object libraries are referenced by the one or more platform-independent programming language objects of the platform-independent programming language representation

Applicant submits that the modified one or more platform-independent programming language objects comprised in the one or more object libraries recited in claim 19 are clearly distinct from the one or more platform-independent programming language objects of the platform-independent programming language representation recited in claim 11, since the former objects are specifically referenced by the latter objects in claim 19. Claims 34 and 52 include similar language. Thus, Applicant asserts that claims 19-23, 34 and 52 are clearly definite under 35 U.S.C. § 112, second paragraph..

Section 103(a) Rejection:

The Office Action rejected claims 11, 13-19, 21, 23-28, 30-38, 40-46 and 48-54 under 35 U.S.C. § 103(a) as being unpatentable over Wang (U.S. Patent 6,292,936) in view of "The IR to VMx86 Translation Module Specification" by Chris Lattner, Dec.

1999 (hereinafter "Lattner"). Applicant respectfully traverses this rejection in light of the following remarks.

The Examiner asserts that Wang teaches accessing a sequence of script language instructions and generating a platform-independent representation of the one or more script language instructions, wherein the platform independent programming language produces results in accordance with the original sequence of script language instructions. The Examiner further asserts that Wang teaches that the platform-independent programming language representation comprises a sequence of logical commands but does not teach that each of the commands is stored as one or more platform-independent programming language objects. However, the Examiner asserts that Lattner teaches representing instructions using a Java 'Instruction' class, and that it would have been obvious to combine Wang and Lattner since programming objects are easier to create and manipulate.

Wang teaches a method for providing an interpreter-base scripting environment that includes multiple runtime processors executed by the computer. (col. 1, lines 40 – 44). Specifically, Wang teaches a server system 106 executing a Web daemon 108 including one or more runtime processors, which may comprise a Java Virtual Machine 110 and a VisualBasic Script interpreter 112. The server system may further include one or more translators 114 that are operable to translate an original input source for the one or more runtime processors. (col. 2, lines 40 – 58). In one embodiment the translator 114 may separate the original input source into two sources, one for the Java Virtual Machine 110 and one for the VisualBasic Script interpreter 112. The translator may also translate every HTML block of the original source into a synchronizer token. Execution flow may then move back and forth between the Java Virtual Machine 110 and the VisualBasic Scrip interpreter 112 according to said tokens. (col. 3, line 25 – col. 4, line 30).

Lattner teaches a system for the translation of an intermediate representation to a target virtual machine. (page 1). Specifically, Lattner teaches an Instruction class which is designed to exist as a node in a linked list and generate a legal 80x86 assembly

language string for the intermediate language instruction associated with that particular node in the list. (page 2).

Lattner teaches the generation of *non-platform-independent assembly code* from a set of Java objects representing intermediate instructions. Each of Lattner's intermediate instructions as encapsulated by the Instruction class is executable only insofar as it can generate platform-specific assembly code. Accordingly, Applicant can find no language in Wang or Lattner that teaches or suggests, either separately or in combination, a method "wherein the platform-independent programming language representation is executable to produce results in accordance with the original sequence of script language instructions," as recited in Applicant's claim 11.

Moreover, Applicant disagrees with the Examiner's assertion that it would have been obvious to combine Wang with Lattner "since programming objects are easier to create and manipulate." As previously stated, Lattner teaches the generation of *non-platform-independent assembly code* from a set of Java objects representing intermediate instructions, while Wang teaches a technique to allow multiple runtime processors to process their corresponding portion of the original input source in a synchronized manner, thereby solving the problem of a container context requiring multiple run-time processors. (col. 1, lines 28 – 34). Lattner does not suggest representing script instructions as executable platform-independent programming objects. The combination of Wang and Lattner does not teach generating a platform-independent programming language representation of the sequence of script language instructions executable to produce results in accordance with the original sequence of script language instructions.

The combination of Wang and Lattner as suggested by the Examiner, comprising "access[ing] a sequence of script language instructions and generat[ing] a platform-independent representation of the one or more script language instructions where the platform independent representation produces equivalent results with the script language instructions, as taught by Wang, where the platform-independent representation includes instructions represented by objects, as taught by Lattner" would run contrary to Wang's

goal of multiple runtime processors, since Lattner teaches the generation of 80x86 assembly language, and hence, only one type of runtime processor. Thus, Applicant can find no reason why it would be obvious to combine Wang with Lattner.

Accordingly, claim 11 is believed to patentably distinguish over the cited art for at least the reason above. Likewise, independent claims 28, 38 and 46 recite subject matter similar to Applicant's claim 11, and are thus believed to patentably distinguish over the cited art for at least the reason given above.

Claims 12, 29, 39 and 47 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Wang in view of "The Principles of Computer Hardware, Third Edition" by Alan Clements, 2000 (hereinafter "Clements"). In rejecting claim 12, the Examiner states that Wang teaches the method of claim 11 but does not teach the limitations of claim 12. However, the Examiner already admitted in the rejection of claim 11 that Wang alone does not teach the method of claim 11. Specifically, the Examiner admitted that Wang does not teach that each of the commands is stored as one or more platform-independent programming language objects. The Examiner does not assert that Clements overcomes this deficiency in Wang. Since claim 12 depends upon claim 11, and claim 11 is not taught by Wang and Clements, the rejection is improper.

Furthermore, claim 12 is further distinguishable over Wang in view of Clements. The Examiner asserts that Clements teaches using the stack data structure to hold instructions that are executed by popping the instruction off the stack, and that it would have been obvious to perform the detecting, generating, interpreting, executing and accessing steps of Claim 11 since this (i.e. use of a stack) is an intuitive way to parse instructions in a computer system.

However, Clemens teaches assembly language programming for the 68K family, specifically a computer *hardware* stack (Clement, page 3) Applicant can find no language in Wang or Clement which teaches or suggests, either separately or in

combination, a "**stack data structure**... wherein said interpreting and executing is performed by **a stack-machine interpreter engine**," as recited in Applicant's claim 12.

Furthermore, if the Examiner makes a new ground of rejection for claim 12 including Lattner, the Examiner should note that Lattner teaches the use of an Instruction class to encapsulate intermediate instructions in a linked list (page 2) while Clement teaches instructions in a hardware stack (page 3). A linked list is a wholly different type of structure and incompatible with a hardware stack, and as such Applicant can see no feasible way, and thus no reason, to combine Wang as modified by Lattner with Clement.

Accordingly, claim 12 is believed to patentably distinguish over the cited art for at least the reason above. Likewise, claims 29, 39 and 47 recite subject matter similar to Applicant's claim 12, and are thus believed to patentably distinguish over the cited art for at least the reason given above.

The Office Action rejected claims 20 and 22 under 35 U.S.C. § 103(a) as being unpatentable over Wang in view of Lattner and further in view of "Load-time Structural Reflection in Java" by Shigeru Chiba, June 2000. Claims 20 and 22 are patentable for at least the reasons given above in regard to claim 11.

Applicant also asserts that numerous ones of the dependent claims recited further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicant submits the application is in condition for allowance, and notice to that effect is requested.

If any extension of time (under 37 C.F.R. § 1.136) is necessary to prevent the above referenced application from becoming abandoned, Applicant hereby petitions for such extension. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-60300/RCK.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

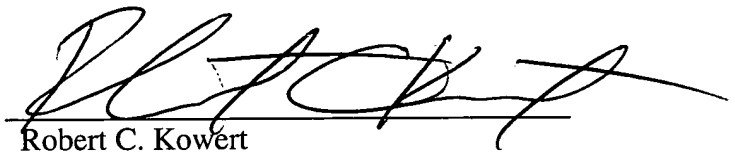
☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Fee Authorization Form authorizing a deposit account debit in the amount of $ for fees ( ).

☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: November 3, 2003